

Adaptively Detecting Malicious Queries in Web Attacks

Ying Dong^a, Yuqing Zhang^{a,*}

^aNational Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing, China

Abstract

Web request query strings (queries), which pass parameters to the referenced resource, are always manipulated by attackers to retrieve sensitive data and even take full control of victim web servers and web applications. However, existing malicious query detection approaches in the current literature cannot cope with changing web attacks with constant detection models. In this paper, we propose AMODS, an adaptive system that periodically updates the detection model to detect the latest unknown attacks. We also propose an adaptive learning strategy, called SVM HYBRID, leveraged by our system to minimize manual work. In the evaluation, an up-to-date detection model is trained on a ten-day query dataset collected from an academic institute's web server logs. Our system outperforms existing web attack detection methods, with an F-value of 94.79% and FP rate of 0.09%. The total number of malicious queries obtained by SVM HYBRID is 2.78 times that by the popular Support Vector Machine Adaptive Learning (SVM AL) method. The malicious queries obtained can be used to update the Web Application Firewall (WAF) signature library.

Keywords: Web attacks, Web request query strings, Adaptive learning, SVM, Intrusion detection, Anomaly detection

1. Introduction

Web attacks are attacks that exclusively use the HTTP/HTTPS protocol. Symantec Internet Security Threat Report (ISTR) [1] reveals that the number of web attacks explosively increased, reaching 1.1 million every day in 2015, more than twice the rate in 2014 (0.493 million). Among web attacks, code injection attacks against web applications [2] increased each year and accounted for at least 96.15% of web attacks in 2015, according to the Imperva Web Application Attack Report (WAAR) [3–5]. In 2015, Team GhostShell claimed to have hacked numerous websites using SQL injection (SQLI) attacks [6], and disclosed thousands of compromised account details, including emails, user names, addresses, telephone numbers, and other privacy information [7].

This paper focuses on the most frequent types of web-based code-injection attacks in 2015 [5], namely, Cross-Site Scripting (XSS) attack [8], SQLI attack, Directory Traversal (DT) [5], and Remote File Inclusion (RFI) [5], respectively accounting for 49.09%, 28.32%, 9.82% and 8.92% of the entire web attacks. Attack vectors of these attacks exist in user input. Whenever user input is improperly handled, these attacks can succeed. Since most user input data exists in queries¹, detecting malicious queries is the

core of detecting web-based code injection attacks. For web applications whose source code is unavailable or difficult to obtain, Web Intrusion Detection Systems (Web IDSs) are the only option [9]. A web IDS acts as an intermediate layer between protected web applications and users, and analyzes web traffic to detect possibly malicious activities.

A considerable amount of effort has been made to detect malicious queries in web requests using web IDSs. Two major detection approaches are utilized: signature-based detection and anomaly-based detection. A popular signature-based detection approach is WAF, which maintains a signature library of known signatures, and compares new web requests with the signatures. Signature-based approaches are efficient in detecting known attacks with a low False Positive (FP) rate, but unable to detect previously unknown attacks (e.g., zero-day attacks). Anomaly-based detection approaches [10–15] usually rely on a detection model to identify anomalous web requests. In contrast to signature-based approaches, anomaly-based approaches can detect unknown attacks, but with a high FP rate. The two approaches are often applied in a complementary fashion in web IDSs: anomaly detection serves to discover unknown attacks, whose attack signatures are then utilized by signature-based approach for detection. The ability to detect unknown attacks has drawn wide academic attention to anomaly-based approaches.

However, existing anomaly-based approaches for web attack detection are constant, in other words, they usually collect all training data once to induce a constant detection model. Since attackers have become more sophisticated and utilized more advanced web attack toolkits, the

*Corresponding author. Tel.: +8601069671800.

Email addresses: dongying115@mails.ucas.ac.cn (Ying Dong), zhangyq@ucas.ac.cn (Yuqing Zhang)

¹Web request query string is consistently termed as query in this paper for simplicity.

detection model might become obsolete and outdated, incapable of detecting the latest malicious queries in web attacks. Previous adaptive attack detection methods [16–18] are designed for network intrusions and are not applicable to web attacks. A practical solution for keeping the web attack detection model constantly updated is to incorporate the latest important queries, including informative benign queries and representative malicious queries. The existing approach to obtaining the latest important queries is randomly selecting requests from web traffic and then manually labeling them. Unfortunately, the majority of randomly selected queries are similar benign ones, and there is low probability of a query being important. Since web traffic is huge, it does take considerable manual labeling work to obtain important queries. This challenge has motivated our research work to build a malicious query detection system that can adaptively incorporate the latest important queries to update the detection model.

In this paper, we address the issues related to the adaptive detection of malicious queries in web attacks. We present AMODS, an adaptive system for this purpose. AMODS leverages an efficient adaptive learning strategy, SVM HYBRID, which is a hybrid of Suspicion Selection (SS) and Exemplar Selection (ES). The former features in acquiring the most important informative queries, namely, suspicions, while the latter specializes in harvesting representative malicious queries, exemplars. Suspicions and exemplars are called important queries. AMODS aims to identify attacks as early as possible by periodically checking the latest important queries. The number of important queries is trivial, so manual labeling work is reduced to the minimum. The important queries are then incorporated into the training pool to update the detection model, which is a stacking-based ensemble classifier, composed of three base classifiers and a meta classifier. SVM [19] is used as its meta classifier, so that SVM HYBRID can exert the detection model to obtain important queries. A ten-day query dataset collected from an academic institute website’s web server logs is used for evaluation. Since the logs record requests passing through a commercial WAF, malicious queries obtained by our system can be used to update the WAF signature library.

The main contributions of this study are as follows:

- We present AMODS, an adaptive system for detecting malicious queries in web attacks. To the best of our knowledge, this is the first work in the adaptive detection of web attacks.
- We propose SVM HYBRID, an adaptive learning strategy designed for the efficient selection of important queries.
- We employ stacking, a meta-learning [20] based ensemble classifier, as the underlying detection model for accurate malicious query detection.
- AMODS outperforms existing web attack detection methods with an F-value of 94.79% and FP-rate of

0.09% on a real-world ten-day query set. The total number of malicious queries obtained by SVM HYBRID is 2.78 times that by SVM AL during the ten-day detection using AMODS.

The remainder of the paper is organized as follows. Section 2 provides background and related work. Detailed design and implementation of the system, together with the proposed adaptive learning strategy, namely, SVM HYBRID, will be presented in Section 3. Section 4 shows data collection and preprocessing. Next, Section 5 introduces the process of the determination of feature reduction and stacked classifiers, then shows the efficacy of our system. In addition, AMODS is compared with constant models and adaptive models, including SVM AL and random selection, in terms of detection performance and the number of malicious queries obtained. Detection performance comparisons with related work are also demonstrated. Afterwards, we clarify some relevant issues of our approach in Section 6. Finally, Section 7 concludes this paper.

2. Background and related work

2.1. Background

Queries contain most user inputs and are thus closely related to web-based code injection attacks, which are the most threatening web-layer attacks [3–5]. Therefore, our work concentrates on malicious query detection in order to detect web-based code injection attacks. Queries can be found in HTTP GET requests recorded in web server logs. Figure 1 shows a web server log entry in Common Log Format (CLF), consisting of various items, such as the source IP address, timestamp, sever response status code, user agent header field, and the actual web request string. The framed text in the request string is a query, which our system examines. A query is identified by a leading “?” character following the referenced resource, and lists pairs of parameter names and values.

Malicious queries in web-based code injection attacks and normal queries are different in character distribution and character sequences. Table 1 compares a normal request and four types of code injection attacks analyzed in this paper: SQLI attack, XSS attack, DT attack, and RFI attack. We include Remote Code Execution (RCE) into XSS attack, since RCE is essentially a special kind of XSS attack [21]. The requests in Table 1 are forwarded to a

```
223.241.162.53 - -
[12/May/2015:23:59:59 +0800]
"GET /resource?parameter1=value1&parameter2=value2
HTTP/1.1" 200 1922
"http://shopping/start.html"
"Mozilla/4.08 [en] (Win98; I ;Nav)"
```

Figure 1: Web server access log entry

Table 1: Comparison of normal queries and malicious queries in web-based code injection attacks

No.	Type	Query in a web request URL
(a)	Normal	http://www.victim.com/index.php?postID=123
(b)	SQLI	http://www.victim.com/index.php?postID='/**/union/**/select/**/0,concat(username,0x3a,password)/**/from/**/users/'
(c)	XSS	http://www.victim.com/index.php?postID=<script>document.location="http://vicious_site/stealcookie.cgi?" + document.cookie</script>
(d)	DT	http://www.victim.com/index.php?postID=../../../../etc/passwd
(e)	RFI	http://www.victim.com/index.php?postID=http://vicious_site/hack.txt?ls

vulnerable script named “index.php” in a forum website, and the script can extract posts from the database and present a post to a user. “postID” is a parameter name for this script and takes an integer input value as the post ID while recording the transaction. As shown in Table 1, (a) “postID=123” is a normal query, and (b)-(e) are malicious queries. (b) is a typical SQLI attack that attempts to obtain confidential data from a restricted table. (c) conducts an XSS attack to obtain user cookies using a Common Gateway Interface (CGI) script on the attacker host. (d) conducts a DT exploit to discover the file “passwd” using string “../../../../”. Since “passwd” is a local file on the vulnerable server, (d) is also called Local File Inclusion (LFI) attack. (e) tries to load “hack.txt” file, a bash shell, from a remote malicious sever, and execute the “ls” command within the bashshell in an RFI attack.

2.2. Related work

Applying anomaly detection to attack detection has been an active research area. Based on whether a detection method can adapt to the change of web attacks, we divide existing web attack and HTTP attack detection techniques that use anomaly detection schemes into two categories: adaptive detection and constant detection. Adaptive detection methods always incorporate new traffic data to induce an up-to-date detection model, while

constant detection methods use a period of traffic data to build a detection model once and use it for all later detection.

Unfortunately, existing adaptive detection techniques [16–18] are designed for network intrusion detection. They might not be suitable for web attack detection, since the landscape of network intrusions differs from web attacks. Malicious queries in web attacks have more subtle distinctions from legitimate queries in normal requests. Constant detection methods [10–14] cannot detect the latest unknown attacks. The adaptive detection of web attacks has been found deficient.

In this paper, AMODS is proposed to bridge the gap between adaptive detection and web attack detection. As far as we know, this is the first research in the adaptive detection of web attacks. Figure 2 gives our taxonomy of the latest research on methods that use anomaly detection to detect web attacks as well as HTTP attacks.

2.2.1. Adaptive detection

Adaptive detection can detect the latest unknown attacks by adapting to the changing behaviors of attacks, yet adaptive detection methods to date have been only related to network intrusion detection.

HTTP attack detection. These detection methods take network traffic of port 80, namely HTTP traffic, as data

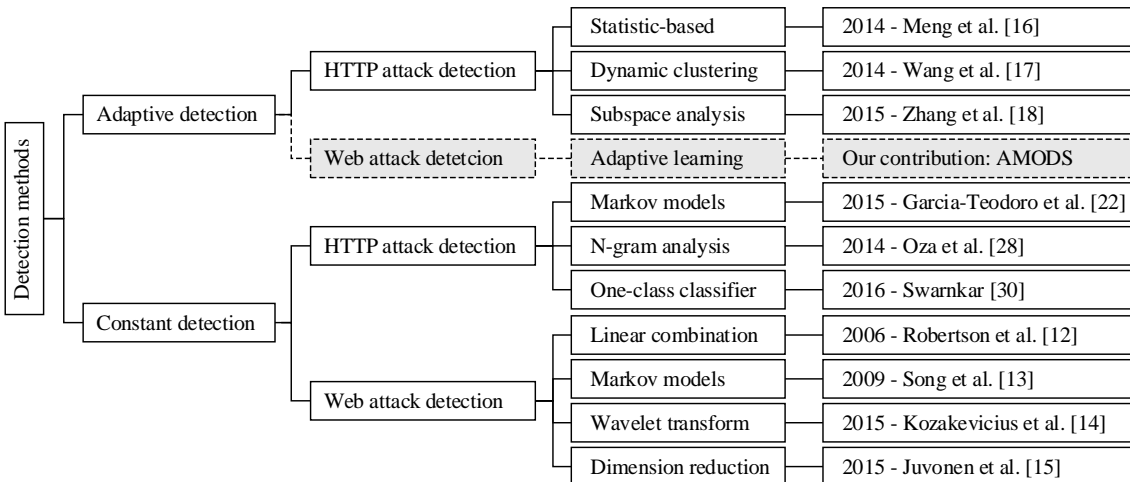


Figure 2: Taxonomy of the latest academic research on anomaly detection methods of web attacks and HTTP attacks

source and analyze HTTP packet payloads. Three types of unsupervised learning techniques are utilized: (a) Statistic based. Meng et al. [16] used a statistic-based approach to develop an adaptive blacklist-based packet filter, which can help filter out network packets based on IP confidence. (b) Dynamic clustering. Wang et al. [17] employed Affinity Propagation (AP) in dynamical clustering to detect HTTP attacks adaptively. (c) Subspace analysis. Zhang et al. [18] proposed Adaptive Stream Projected Outlier Detector (A-SPOT), which used a novel adaptive subspace analysis approach to detect anomalies from network connections.

Our proposed system AMODS falls into the category of adaptive methods for web attack detection. AMODS leverages an efficient adaptive learning strategy, called SVM HYBRID. Queries obtained by SVM HYBRID are used to update the detection model periodically. AMODS can identify attacks at an early stage with a high F-value, and significantly improve the detection performance of the model over time.

2.2.2. Constant detection

Constant detection methods are widely applied to intrusion detection. Though effective, they suffer from a high F-P rate. To deal with this problem, we combined SVM HYBRID with a stacking-based detection model. Using our detection model, we achieved the lowest FP rate in comparison with above constant detection methods as shown in Section 5.4. Constant detection methods for HTTP attacks and web attacks are summarized as follows.

HTTP attack detection. Constant detection methods of HTTP attacks model packet payload in HTTP traffic, and operate at packet layer. Four categories of detection algorithms are used: (a) Markov models. Garcia-Teodoro et al. [22] represented the normal behaviors of a service by Markov models, aiming to generate HTTP intrusion signatures for Network Intrusion Detection Systems (NIDS). Soon, Zhong et al. [23] used Markov models whose inputs are each character of parameter values in normal payloads to build the normal profile. Earlier works, such as HMM-Payl [24], made similar use of Markov models. (b) N-gram analysis. N-gram is a text categorization technique and is widely used in NIDSs, such as PAYL [25], Anagram [26], RePIDS [27] and [28]. (c) One-class classifier. McPAD [29] detects HTTP attacks using multiple one-class SVMs, whose outputs are combined to make a final prediction. Recently, Swarnkar [30] adopted a one-class Multinomial Naive Bayes classifier and used likelihood of each short sequences occurrence in payloads of known benign packets as a measure to derive the degree of maliciousness of a packet.

Web attack detection. Constant detection methods of web-layer code injection attacks usually analyze queries in web traffic (web server logs), and operate at application layer. Two detection methods are mainly used: (a) Linear combination. Linear combination method creates a profile for normal traffic and considers new web requests

with certain noticeable deviations from the normal profile as attacks. Kruegel and Vigna [10, 11] were the first to analyze web server logs to detect attacks against web servers and web applications. They calculated the final anomaly score for each web request using the linear combination of six models, such as the character distribution of query parameters. Robertson et al. [12] used similar models to generalize suspicious web requests into anomaly signatures. (b) Markov models. Spectrogram [13] used a linear mixture of multiple Markov-chains to model queries. (c) Wavelet transform. Kozakevicius et al. [14] proposed a URL query string anomaly sensor based on the properties of the bidimensional Haar wavelet transform. (d) Dimensionality reduction. Juvonen et al. [15] proposed an anomaly detection system to detect anomalies in queries, and used dimensionality methods for data preprocessing.

It is worth mentioning that we only concentrate on adaptive detection of web attacks; two related problems, namely vulnerability exploitation and signature generation, are beyond the scope of this paper. On the one hand, the criterion of query labeling is a combination of signature-based method and manual analysis, as described in Section 6, regardless of whether the corresponding requests successfully exploit the vulnerabilities of an application or not. Web application vulnerability exploitation detection can be done via either static analysis [31] or dynamic analysis, such as taint tracking [32] and taint inference [33]. On the other hand, we simply expect to obtain as many malicious queries as possible to update the WAF signature library. Signature generation is outside our subject, which commonly uses regular expressions [34].

3. Design and implementation

In this section, we first illustrate the design of our Adaptive Malicious Query Detection System (AMODS, “O” for “Q”), followed by the core adaptive learning technique, named SVM HYBRID. Lastly, we introduce our stacking-based training and classification.

3.1. System design

In this paper, we introduce the concept of *important queries*, which consist of the most informative normal queries and the most representative malicious queries, as mentioned in Section 1. The goal of AMODS is to identify attacks as early as possible by periodically checking the latest important queries. To reach this goal, our system leverages an adaptive learning approach called SVM HYBRID to obtain important queries.

The system is initialized with a training pool composed of a small set of labeled queries, from which the initial detection model is trained. The detection model is based on stacking, and composed of three base classifiers and a meta classifier. SVM is used as its meta classifier, so that SVM HYBRID could exert the detection model to obtain important queries. To update the initial detection model,

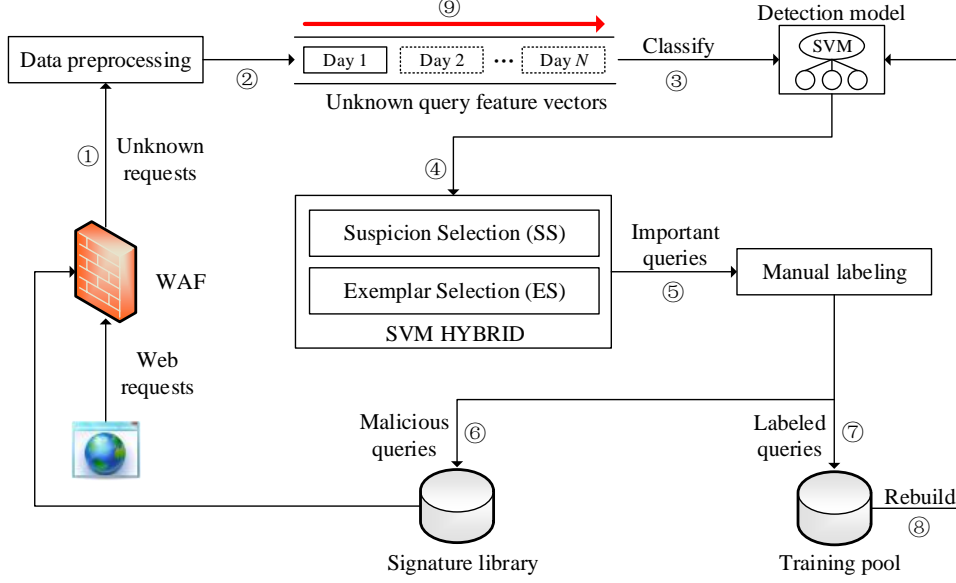


Figure 3: System overview

the system takes web server logs as its data source, and examines queries in HTTP GET requests in batch mode. These queries are called unknown queries, since their class labels are unknown. Classified unknown queries are analyzed by SVM HYBRID to obtain the smallest number of important queries for labeling. Thus, the manual work is greatly reduced. As shown in Figure 3, the system proceeds as follows.

- (1) Collecting unknown queries. Unknown queries are collected from N -day web requests. Since attacks in the web requests have been filtered out by WAFs, the collected unknown queries contain benign queries and malicious queries undetected by WAFs.
- (2) Data preprocessing (Section 4.2). Data preprocessing transforms raw queries into feature vectors in batches of N days, including data preparation, feature construction, and feature reduction.
- (3) Classification (Section 3.3). Unknown queries on the 1st day are classified using the current detection model.
- (4) Queries selection (Section 3.2). A fixed number of important queries are obtained from classified unknown queries on the 1st day using SVM HYBRID.
- (5) Label the obtained important queries manually.
- (6) Update the WAF signature library using the obtained important queries that are labeled as malicious.
- (7) Add all the labeled important queries into the training pool².
- (8) Update the detection model (Section 3.3).
- (9) Iterate steps (3)-(8) for unknown queries in the rest $N - 1$ days.

²In following text, the terms “training pool”, “training set”, and “labeled set” are used interchangeably.

3.2. SVM HYBRID

SVM HYBRID, our proposed adaptive learning strategy, obtains important queries to update the malicious query detection model. SVM HYBRID operates in the kernel feature space of SVM (Support Vector Machine), and relies on positions of query samples in SVM kernel feature space to decide queries for labeling. SVM has performed extremely well in many domains, particularly those involving text classification.

Given a domain X , a linear classifier used to separate positive and negative samples [19] is defined in terms of a hyperplane

$$w^T x + b = 0, (w \in R^N, b \in R), \quad (1)$$

where w is the orthogonal normal vector to the hyperplane and is assumed to point towards the positive class, and b is the bias of the hyperplane. The linear classifier is corresponding to the decision rule

$$y = \text{sgn}(w^T x + b). \quad (2)$$

Given training set $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where $x_i \in R^N$ is a feature vector, and $y_i \in \{-1, 1\}$ is a class label, the margin is defined as the maximal perpendicular distance between samples of the two classes. SVM aims to find the optimal linear classifier whose separating hyperplane maximizes the margin. w of the optimal linear classifier is solved as

$$w = \sum_{i=1}^n a_i y_i x_i. \quad (3)$$

When original input samples are non-linearly separable, the samples could be mapped to a new kernel feature space where they are linearly separable. Given a feature

mapping process φ , the corresponding kernel is defined as

$$K(x, z) = \varphi(x)^T \varphi(z). \quad (4)$$

Using Equation (3) and (4), the decision value in Equation (2) can be rewritten as:

$$f(x) = \sum_{i=1}^n \alpha_i y_i K(x_i, x) + b. \quad (5)$$

$f(x)$ is the kernel distance between a sample x and the hyperplane. Calculating $K(x, z)$ is usually trivial. Besides, according to the Karush-Kuhn-Tucker (KKT) conditions, α_i is non-zero only for samples that are closet to the hyperplane, namely support vectors, the number of which is usually small. Therefore, the kernel distance is efficient to compute.

For data that are non-linearly separable though mapped to infinite dimensional feature space, the optimization problem can be formulated as

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, i = 1, \dots, n \\ & \sum_{i=1}^n \alpha_i y_i = 0, \end{aligned} \quad (6)$$

where penalty parameter C controls the relative weighting of a large margin and low misclassification rate of training data. A small C prefers a large margin, at the cost of a large number of misclassified training samples, corresponding to a soft margin classifier. By contrast, a large C means a hard margin classifier.

The illustrative graph of SVM HYBRID is presented in Figure 4. The middle solid line is the separating hyperplane, while the two external solid lines indicate the margin. The upper side of the hyperplane is the malicious side. “+” represents a query in the positive class, namely, a malicious query, while “-” stands for a query in the negative class: a normal query. SVM HYBRID comprises of Suspicion Selection (SS) and Exemplar Selection (ES). *SS obtains the most important informative queries*, which are suspected of being malicious and thus are called suspicions. Suspicions are within the prone-to-be-misclassified region, called the confusing region: the grey area delimited by two dashed lines in Figure 4. The two points surrounded by red circles within the confusing region are suspicions. Conversely, *ES obtains the most representative malicious queries*, which are called exemplars, since each of them is an exemplar of similar malicious queries. Exemplars lie in the malicious side and far away from the hyperplane. As shown in Figure 4, the four points surrounded by red circles on the upper side are exemplars. The main contribution of suspicions and exemplars is also different: the former contributes more to detection performance improvement, while the latter is conducive more

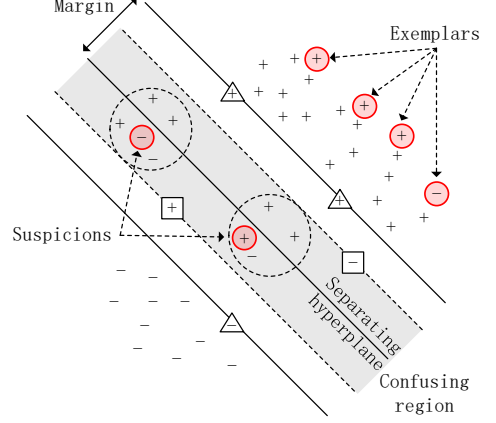


Figure 4: Illustrative graph of SVM HYBRID

to harvesting truly malicious queries. The contribution of SVM HYBRID is a hybrid of these strengths.

SS and ES operate on two disjointed subsets of each unknown query set. SVM HYBRID starts with SS on one subset and then switches to ES on the other subset. The ratio of the size of query subset for SS to that for ES is denoted as θ . θ is an adjustable parameter in SVM HYBRID to address the tradeoff between detection performance improvement and the number of malicious queries obtained. The effects of θ set to different values are evaluated in Experiment 2 (Section 5.1).

Moreover, SVM HYBRID is suitable for SVMs with any kernels. Radial basis function (RBF) kernel and Polynomial kernel, two widely used kernels, are tested in our experiment, respectively defined as

$$K(x, y) = e^{-\gamma \|x - y\|^2} \quad (7)$$

$$K(x, y) = ((x \cdot y) + \theta)^d. \quad (8)$$

3.2.1. Suspicion Selection (SS)

Adaptive learning with SVM uncertainty sampling is also known as SVM active learning (we call it SVM AL in the rest of this paper) [35, 36]. SS is similar to SVM AL, since they all select unlabeled samples within the margin. However, SVM AL simply selects samples closet to the hyperplane and does not consider their distribution. Consequently, similar samples might be selected, making SVM AL prone to sample bias. Labeling similar samples is also a waste of manual work. Therefore, given a large pool of unlabeled samples, it is desirable to preserve the density distribution of samples in the pool during the learning process.

Our proposed Suspicion Selection (SS) pursues the idea of preserving data distribution by using clustering to find potential informative queries from unknown queries within the confusing region. In brief, SVM AL concentrates on the most informative data, while SS focuses on the most important informative data so that data redundancy is reduced to the minimum. The procedure of SS is as follows.

Firstly, the confusing region is determined. The set of misclassified queries in a training set that fall into the margin during training is denoted as Q . Queries in Q are sorted in order according to their kernel distances computed using Equation (5). The query in Q with the minimal kernel distance (the “+” point with the rectangular frame in Figure 4) determines the lower boundary of the confusing region (the lower dashed line). Oppositely, the query in Q with the maximal kernel distance (the “-” point with the rectangular frame) determines the upper boundary of the confusing region (the upper dashed line). That is,

$$f_{lower} = \min_{x \in Q} f(x) \quad (9)$$

$$f_{upper} = \max_{x \in Q} f(x). \quad (10)$$

The confusing region is defined as

$$Confusing\ region \Leftarrow f_{lower} \leq f(x) \leq f_{upper}. \quad (11)$$

To ensure the existence of the confusing region, the margin should be wide enough to allow misclassifications of training samples within the margin. That is, the meta classifier SVM of our detection model should be a soft margin classifier. The training misclassification rate and detection performance of the detection model can be balanced by adjusting the penalty parameter C and SVM kernel parameter of the meta classifier SVM.

Secondly, K -medoids clustering [37] is performed on the unknown queries that fall into the confusing region. The final cluster centers are suspicions. K -medoids begins with K randomly selected samples as medoids (cluster centers). Then each remaining sample is assigned to the cluster whose medoid is the closet to it. That is, the partitioning method is based on the principle of minimizing the sum of the distances between each sample and its corresponding referenced medoid. This sum is denoted as E , and is defined as

$$E = \sum_{j=1}^k \sum_{x \in C_j} |\varphi(x) - \varphi(M_j)|, \quad (12)$$

where x is a given sample in cluster C_j , whose cluster center is M_j . $|\varphi(x) - \varphi(M_j)|$ is the Euclidean distance between x and M_j in kernel feature space, and using Equation (4), it can be given by

$$\begin{aligned} & |\varphi(x) - \varphi(M_j)| \\ &= \sqrt{K(x, x) + K(M_j, M_j) - 2K(x, M_j)}. \end{aligned} \quad (13)$$

The K cluster centers with which E reaches its minimum value are optimal. The partitioning step is repeated until the number of unknown samples within the confusing region is less than K . When partitioning terminates, the final cluster centers are the most important informative samples, called suspicions. Suspicions are important

samples whose class labels cause the most confusion to the classifier.

Figure 4 shows that the confusing region covers two clusters, each represented by a dashed circle. Their cluster centers are the two points surrounded by red circles in the two dashed circles. One cluster center is a normal query and another is a misclassified malicious query. In other words, suspicions might be misclassified and also might not be malicious. On the contrary, queries obtained by Exemplar Selection (ES) are highly likely to be malicious. ES is described in the following subsection.

3.2.2. Exemplar Selection (ES)

Exemplar Selection (ES) serves to obtain representative malicious queries, called exemplars, from the unknown queries. New web attacks are very commonly variants of existing attacks. Queries in variant attacks might be similar to queries in existing attacks. Labeling similar queries would unnecessarily increase manual labeling workload. As a result, we hope that each exemplar is the unknown malicious query that differs most from all the malicious queries in the current labeled set.

To attain this goal, we apply the Kernel Farthest-First (KFF) algorithm [38], a greedy heuristic, to obtain exemplars from the malicious side of the SVM hyperplane. KFF always chooses from the unlabeled query set the query that is the farthest from all the malicious queries in the labeled set. Given unlabeled query set U and labeled query set L , the farthest distance is defined as the maximum of the sum of distances between each malicious query y in L and a query x in U in kernel feature space. Formally, the optimal x is determined by

$$\arg \max_{x \in U} < \sum_{y \in L} |\varphi(x) - \varphi(y)| >, \quad (14)$$

where $|\varphi(x) - \varphi(y)|$ is computed using Equation (13). Each optimal x is an exemplar, and is added to the labeled set. As can be seen from Figure 4, ES obtains four exemplars, represented by four points surrounded by red circles. One of the exemplars is a misclassified normal query. This is unsurprising, since misclassification might happen. Nevertheless, most exemplars are still truly malicious. At each iteration of the adaptive detection process, suspicions and exemplars are manually labeled and used for refining the decision boundary of the meta classifier SVM of the detection model.

Specifically, SS uses the kernel distance to the hyperplane, which is already solved when a sample is classified by a SVM, while ES uses the kernel distance to malicious queries in current training set, which could be computed using the efficient-to-compute kernel function. Thus SVM HYBRID is applicable for real-time attack detection scenarios.

3.3. Training and classification

In machine learning, learning bias refers to any preference for choosing one hypothesis that explains the da-

Table 2: Description of web server logs and data preprocessing

Item	Raw logs		Data preprocessing			
	Duration (days)	Log size (GB)	Original requests	Cleaned queries	Normalized queries	Filtered queries
#	10	6.11	476,459	188,290	118,928	112,397

Table 3: Statistics of query dataset

#Queries	Initial set	Ten unknown sets
Benign	80	99,080
Malicious	20	920
Total	100	100,000

ta over other equally acceptable hypotheses [39]. Meta-learning is the ability to learn from different learning biases that are appropriate for a particular problem by enriching the model hypothesis space [40]. Meta-learning is also an ensemble classifier technique [41]. Ensemble classifier techniques proved to be effective in intrusion detection, such as HMMPayl [24] and McPAD [29]. The ability to learn from different learning biases makes meta-learning an appropriate approach for malicious query detection, due to the complex, dynamic, confrontational nature of its problem domain.

In our approach, we employ stacked generalization [42], as well as named stacking, a well-known meta-learning technique, as our underlying detection model. The key idea of stacking is to train a meta classifier from the prediction results of base classifiers [42]. By using a meta classifier, stacking tries to induce which base classifiers are reliable and which are not during training. During classification, a test sample is classified by each of the base classifiers; then these classifications are fed into the meta classifier that makes the final decision. The base classifiers in our stacking detection model are chosen from three different classifier families. This combination scheme of base classifiers serves to obtain learning diversity and overcomes the learning bias of similar classifiers in the same category. The meta classifier in our stacking detection model is SVM so that SVM HYBRID can operate on our stacking detection model by exerting its meta classifier SVM.

4. Data collection and preprocessing

4.1. Data collection

Web server logs are provided by most real-world web servers. The availability of this data source is a great advantage of our analysis. The web server logs of an academic institute during the period of July 1-10, 2016 are used to evaluate our system. The institute uses an Apache web server, whose logs are in CLF format as shown in Figure 1. The web application chosen from the institute’s server offers several resources on its portal and consists

of both static and dynamic pages. In addition, this web application is equipped with a commercial WAF, called Hillstone [43], and only requests passing through the WAF are recorded in its web server access logs.

A description of the raw logs and data preprocessing procedure is presented in Table 2. The number of original requests, queries after data cleaning, data normalization and application of character filter are shown. The remaining 112,397 queries are used for our analysis.

The remaining queries are labeled by means of the signature set of ModSecurity [44] and manual analysis, as described in Section 6. ModSecurity is an open-source WAF, used to protect Apache web servers from web attacks. The signatures of ModSecurity are provided by the OWASP ModSecurity Core Rule Set (CRS) [45] project; the CRS version we use is 2.2.9. The reason of using ModSecurity signatures to identify attacks is its relatively better detection performance compared with other WAFs [46]. It is worth noting that we only include the malicious queries of the top four most threatening attacks in our dataset, namely XSS, SQLI, RFI and DT attacks, as noted in Section 2.

Web application concept drift occurred on the academic website within reason during the ten-day experimental period, as described in Section 6. In order to accurately evaluate the performance of our adaptive detection method, we need to alleviate the effect of concept drift on our system. Therefore, normal queries during the ten days are mixed up and then equally divided into ten normal query sets.

Eleven disjointed query sets are used in our experiments, including an initial query set and ten unknown query sets. The initial set is used to induce the initial detection model, while each unknown set is used to update the detection model. The eleven sets are created from the entire 112,397 queries during the ten days. Each unknown query set is composed of 10,000 queries on one day sequentially, and 12,397 queries are left. The number of malicious queries in the ten unknown query sets is imbalanced: 95, 112, 97, 100, 107, 96, 92, 99, 127 and 108. To make the number of malicious queries in each unknown set equal, we use the minimum number of 92 on the 7th day as the number of malicious queries in each unknown set. Then we replace surplus malicious queries in other nine unknown sets with normal ones in the remaining 12,397 queries. These queries are then used to create the initial query set. The detailed statistics of the eleven sets of queries are presented in Table 3.

4.2. Data preprocessing

4.2.1. Data preparation

Our system models queries in HTTP GET requests shown in Figure 1. To collect our query dataset, the following steps are employed.

Data cleaning. Web server logs are examined to collect successful GET requests: requests whose return code is equal to or greater than 200, and less than 300. Then, static requests (e.g., .html, .wav, .txt, .jpg) are removed. Finally, the remaining successful GET requests are parsed to extract queries like *parameter1 = value1¶meter2 = value2* in Figure 1.

Data normalization. Data normalization helps tighten the input space, including decoding printable ASCII characters, un-escaping, transforming to lowercase and removing queries whose length is less than four. After normalization, initially different queries may appear identical. The identical ones show as one entry in our dataset.

Character filter. The general syntax of URI in RFC 2616 [47] defines the unsafe characters that should not appear in standard queries. We consider queries that contain any of the unsafe characters to be malicious and propose a character filter to remove these queries. The unsafe characters defined in RFC 2616 are as follows:

- (1) Any ASCII control character (octets 0-31, 128-255) and DEL (127)
- (2) ASCII SP (32)
- (3) ", #, %, <, >

4.2.2. Feature construction

Feature construction transforms each query into a numerical feature vector of N -grams. An N -gram is an N -character slice of a string [48], and its analysis has been successfully applied to intrusion detection [25–28]. We use N -gram analysis in feature extraction, taking advantage of its two properties. N -gram analysis captures the character distribution and sequence characteristics of a string, and using these, it is able to distinguish malicious queries from normal ones. Furthermore, N -gram analysis is fully automatic and requires no prior knowledge about target web application and target attacks.

The value of N is important in N -gram analysis. When N -gram ($N \geq 1$) features are extracted, the input feature space of labeled query set is defined as

$$S = \{N\text{-gram}_i | i = 1, \dots, 63^N\}. \quad (15)$$

The size of S is 63^N , since 63 unique characters exist in all the queries after data preparation. Intuitively, sufficiently high-order N -grams must be used to obtain more accurate features. However, for $N=1$, N -gram features only capture the character distribution information and lacks the sequence information, which is necessary for attack detection in strings. For $N=2$, the character sequence information is also included. For $N>2$, the number of N -grams exponentially increases with N . The smallest number of features is already 25,507 (63^3), which exacts a high

cost in time and computational complexity. Therefore, we choose $N=2$ to balance computational consumption and detection performance, with only 3,969 (63^2) distinct N -grams used. 65,536 (256^2) N -grams would be involved in the feature space without data normalization and character filter. It is observable that the size of feature space decreases by one order of magnitude after the utilization of data normalization and character filter.

Two varieties of N -gram models are binary-based and frequency-based. To capture the structure of queries more accurately, a frequency-based N -gram model is applied. Each N -gram frequency is then divided by the most frequently appearing N -gram frequency in the same query.

4.2.3. Feature reduction

Feature selection serves to remove from the feature set the N -grams that do not appear in all the queries, and identify N -grams that aid classification.

The number of useful N -grams obtained by feature selection is around 1,000, which is still a large number. A large feature set is impractical in our learning task, since our query dataset is huge. In addition, vast features usually do not enhance classification performance and sometimes even degrade it. Hence, to alleviate the curse of dimensionality, dimensionality reduction is implemented after feature selection.

Juvonen et al. [15] also applied dimensionality reduction to the feature space of queries. Our approach differs from their work in that we apply feature selection before dimensionality reduction in order to reduce dimensionality reduction and training time, as well as improve classification performance.

5. Experiments and evaluation

In this section, we report and analyze the results of a series of experiments. Experiment 1 (Section 5.1) is conducted to determine the best setting of feature reduction and classifiers used for our stacking-based detection model. Experiment 2 (Section 5.2) observes the efficacy of SVM HYBRID in AMODS, with respect to detection performance and the number of malicious queries obtained, while AMODS is compared with constant models (SVM model and stacking model) and other adaptive models (adaptive SVM model, SS, ES, SVM AL and random selection) in Experiment 3 (Section 5.3). Lastly, Experiment 4 (Section 5.4) compares AMODS with other work re the detection performance on our dataset.

All the experiments are implemented in Java and conducted on a machine with 32-GB memory and Inter® Core™ 2.93-GHz CPU. To allow SVM HYBRID to work well with the meta classifier SVM of the detection model, we implement stacking on our own. Other classifiers are implemented using WEKA [49], except for SVM, which is implemented using LibSVM [50].

To evaluate the classification performance of our detection model, the primary metric employed is F-value, an

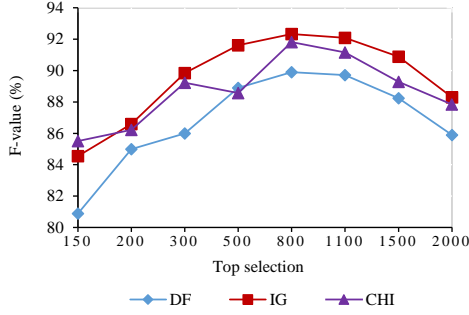


Figure 5: Comparisons of various combinations of feature selection methods and top selection

effective measure for the data imbalance problem. As defined in (16), F-value is a combination of Precision and Recall. Precision measures the percentage of correctly detected malicious queries in all detected malicious queries, while Recall shows the percentage of correctly detected malicious queries in all truly malicious queries. β corresponds to the relative importance of Precision versus Recall and is usually set to 1, as in [27].

$$F\text{-value} = (1 + \beta)^2 * \frac{Precision}{\beta^2 * (Precision + Recall)} \quad (16)$$

In order to comprehensively assess the performance of the detection model, True Positive (TP) rate and FP rate are also utilized.

5.1. Experiment 1: determination of the best setting of feature reduction and stacked classifiers

Experiment 1 aims to determine the best setting of feature reduction and stacked classifiers. The superset of the eleven query sets shown in Table 3 is used. The superset consists of 100,100 queries. Queries in the superset are represented by N -gram feature vectors in the 3,969-dimensional feature space introduced in Section 4.2.2. The best combination of feature selection method and top selection, dimensionality reduction method, and reservation quantity is separately determined on two SVMs, one of which uses RBF kernel, and the other uses Polynomial kernel. Additionally, base classifiers and meta classifier SVM (RBF and Polynomial kernel) of our stacking-based detection model are also determined. These experiments are implemented on the query superset using 10-fold cross validation, amounting to 768 runs in total. The determination metric is the average F-value of each setting. Note that even though the best setting of feature reduction methods and stacked classifiers may vary among different query datasets, the determination scheme can be universally applied to them.

Feature selection and top selection. Various combinations of feature selection methods (Chi-square test, Information Gain (IG) or Document Frequency (DF)) and top selection (150, 200, 300, 500, 800, 1100, 1500 or 2000) are

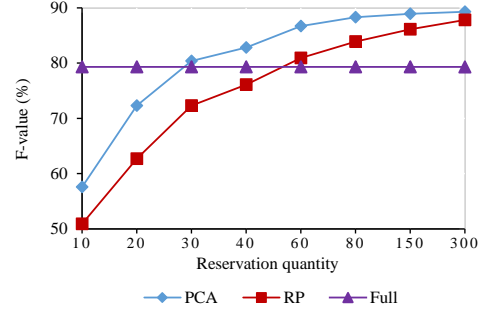


Figure 6: Comparisons of various combinations of dimensionality reduction methods and reservation quantity

compared in Figure 5. It shows that among the three feature selection methods, IG performs best. IG and top 800 overwhelm other combinations, and thus are chosen as the best setting.

Dimensionality reduction and reservation quantity. Different combinations of dimensionality reduction methods (Principal Component analysis [PCA] or Random Projection [RP]) and reservation quantities (10, 20, 30, 40, 60, 80, 150 or 300), as well as a full feature set, are compared in Figure 6. As demonstrated, PCA outperforms RP at lower dimensions, and both of them gradually stabilize when dimensions increase. PCA surpasses the full feature set at the reservation quantity of 80. PCA and 80 reserved dimensionalities are determined as the optimal setting.

Classifier selection. Using the best setting of feature selection and dimensionality reduction, classifier selection is carried out in three steps as shown in Table 4. (a) *Determine base classifier candidates.* Twelve single classifiers are tested, which are chosen from five classifier families [51]: neural networks, Bayesian, tree based, rule based, and regression based classifiers. It can be observed from Table 4 that the performance of single classifiers in different families varies greatly. Even though Bayesian classifiers perform well in the case of the training set, they achieve low F-values of about 60% in the case of cross validation. MLP, Random Forest, JRip and SimpleLogistic perform best in their respective family in both cases. Hence, these four classifiers are chosen as base classifier candidates. (b) *Determine the best combination of base classifiers.* All combinations of three base classifier candidates, together with the combination of all the base classifier candidates are tested in stacking with the meta classifier being SVM with RBF kernel and Polynomial kernel respectively. Table 4 indicates that the combination of Random Forest, Logistic and MLP outperforms not only other combinations but also the best single classifier, namely, Random Forest. Therefore, this combination of base classifiers is used in our stacking-based detection model. (c) *Determine the best meta classifier SVM.* Table 4 shows RBF kernel outperforms the Polynomial kernel, thus SVM with RBF kernel is determined as the meta classifier of the detection model. The optimal pair of SVM penalty param-

Table 4: Determining base classifiers and meta classifier for stacking-based detection model

Classifier type	Classifiers	F-value on	F-value	TP rate	FP rate
		training set(%)	In 10-fold cross validation (%)		
Single classifiers	MLP	85.61	83.77	86.96	0.19
	RBF networks	31.54	23.54	98.91	5.95
	BayesNet	99.87	62.90	96.74	1.03
	Naive Bayes	100.00	60.14	96.74	1.16
	Random Forest	93.17	91.92	98.90	0.15
	J48graft	92.65	91.28	96.73	0.23
	LMT	91.50	90.91	97.83	0.16
	JRip	84.10	81.08	97.89	0.40
	Decision table	57.43	52.49	85.77	1.31
	Conjunctive rule	59.98	54.37	97.83	1.50
	SimpleLogistic	92.92	90.10	98.91	0.19
	Logistic	77.94	76.50	76.09	0.21
Base classifiers	RF SL MLP	96.26	94.30	98.91	0.10
	RF SL JRip	93.87	93.26	97.83	0.11
	RF JRip MLP	92.98	89.11	97.83	0.20
	MLP SL JRip	92.13	90.09	98.91	0.19
	MLP SL JRip RF	94.37	92.32	98.24	0.16
Meta classifiers	SVM-RBF	94.21	93.23	98.99	0.10
	SVM-Poly	93.63	91.19	97.96	0.19

Table 5: SVM HYBRID parameters

Parameter	Adjustable					Fixed	
	θ					M	R
Case	A	B	C	D	E	A-E	A-E
Value	5:5	6:4	7:3	8:2	9:1	150	5

eter C and RBF kernel parameter γ is $(C, \gamma) = (0.05, 2)$, obtained using grid-search cross-validation [52].

5.2. Experiment 2: evaluation of AMODS

Experiment 2 reports and discusses the efficacy of our system, specifically, the impacts of SVM HYBRID on both the detection performance and the number of malicious queries obtained, by adjusting θ (the ratio of the size of query subset for SS to that for ES), which is crucial for SVM HYBRID.

The process introduced in Section 3.1 is carried out using different values of θ , including 1:9, 2:8, 3:7, 4:6, 5:5, 6:4, 7:3, 8:2 and 9:1. In the following, we examine detailed performances of the last five cases, which are denoted as A, B, C, D and E respectively, and whose parameters are listed in Table 5. Two fixed parameters are involved. The number of obtained important queries (suspicions and exemplars) on each day, denoted as M – which should be larger than the number of malicious queries in each unknown set (92) – is set to 150. In addition, the average size of clusters in K -medoids clustering in SS, denoted as R , is set to 5, a quantity we consider appropriate. Then the number of clusters, namely K , is solved as the largest

Table 6: Comparisons of ratios (μ)

Day	μ for various cases				
	A	B	C	D	E
1	2.0	1.5	1.1	0.9	0.7
2	3.4	2.8	2.2	1.8	1.5
3	6.1	5.0	4.0	3.4	3.0
4	11.5	9.0	7.8	6.9	5.8
5	24.0	20.4	17.8	15.7	14.0
6	49.0	36.5	29.0	29.0	24.0
7	74.0	49.0	74.0	36.5	36.5
8	74.0	74.0	49.0	49.0	49.0
9	149.0	74.0	74.0	74.0	49.0
10	149.0	149.0	149.0	74.0	74.0

integer value of the number of unknown queries in the confusing region divided by R .

The ratio of exemplar quantity to suspicion quantity is denoted as μ . Values of μ of the five cases during the ten-day detection are shown in Table 6. As is evident from Table 6, all cases show an increasing trend. This phenomenon is due to the decreasing number of suspicions over time. Exemplars are incremented to make up the reduction in suspicions, so that the total of suspicions and exemplars can stay constant. It can be concluded that ES becomes increasingly dominant in SVM HYBRID during the adaptive detection process.

F-values of the five cases are compared in Figure 7, which shows that all the cases present an increasing trend. For the first three days, relatively low F-values are shown for all cases. At the beginning only a few queries are in

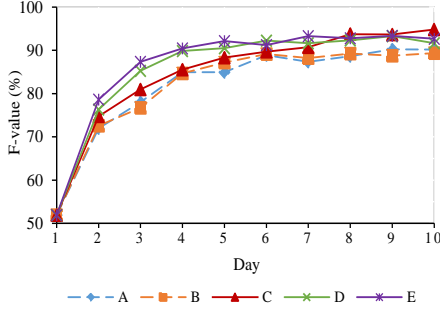


Figure 7: Comparisons of F-values

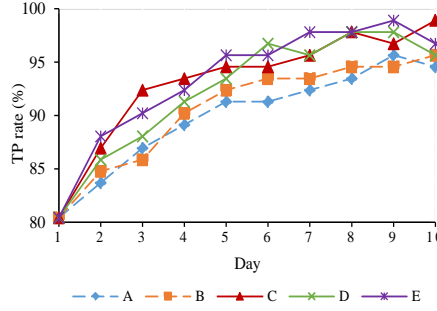


Figure 8: Comparisons of TP rates

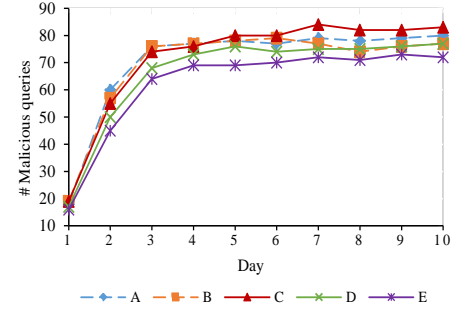


Figure 9: #Malicious queries obtained

Table 7: Comparisons of FP rates

Day	FP rates (%) for various cases				
	A	B	C	D	E
1	1.20	1.20	1.20	1.20	1.20
2	0.45	0.45	0.42	0.36	0.33
3	0.33	0.35	0.33	0.17	0.15
4	0.19	0.21	0.23	0.11	0.11
5	0.19	0.18	0.18	0.12	0.11
6	0.13	0.15	0.15	0.12	0.13
7	0.15	0.17	0.14	0.12	0.11
8	0.16	0.16	0.10	0.13	0.12
9	0.15	0.17	0.09	0.11	0.12
10	0.17	0.17	0.09	0.12	0.11

Table 8: The process of case C

Day	Number of queries			
	Margin	Conf.	Suspicious	Exemplars
1	450	351	70	80
2	339	237	47	103
3	257	151	29	121
4	203	87	17	133
5	152	41	8	142
6	131	24	4	146
7	114	16	3	149
8	98	14	2	148
9	85	10	2	148
10	59	8	1	149

the training pool, so the decision boundary of the detection model tends to be inaccurate. With more suspicions and exemplars incorporated into the training pool, the detection performance is rapidly improved. During the ten days, the trends of cases C, D and E always appear above that of A and B. Recall that θ s of C, D and E are larger than that of A and B, as shown in Table 5; thus C, D and E obtain more suspicions. It can be inferred that suspicions contribute more to detection performance in comparison with exemplars. This is reasonable because the updated detection model would make more accurate prediction of new queries, when the most important uncertain (informative) queries (suspicions) are incorporated into the training pool, rather than representative certain queries (exemplars).

As for cases C, D and E, the trend of case C is naturally expected to be below those of D and E, since more suspicions mean better detection performance, as just inferred. However, case C reverses this expectation: its increasing trend is slightly steeper than those of D and E, and exceeds them on the 8th day. A closer inspection of the three cases reveals that case C obtains the largest total number of misclassified suspicions before the 8th day among the three cases. We attribute the superiority of case C to the misclassified suspicions it obtains, because misclassified suspicions are more informative than other correctly-classified queries. As a result, incorporating them contributes to a more accurate decision boundary of the detection model.

el. On the last day, case C achieves the highest F-value (94.79%) among the five cases during the ten days, which implies the potential of case C to improve detection performance.

Figure 8 and Table 7 exhibit TP rates and FP rates of the five cases respectively. Figure 8 explicitly shows that their TP rates all gradually increase as days go on, while Table 7 suggests that their FP rates rapidly decline for the first three days and then gradually stabilize for the duration. On the 10th day, the TP rates and FP rates of the five cases fall in the ranges of [95.65%,98.91%] and [0.09%,0.17%] separately. On this day, case C achieves the highest TP rate and lowest FP rate of the ten days among all the cases.

As analyzed above, case C outperforms other cases. Since θ is 7:3 in case C, the each-day unknown subset for SS and ES in case C consists of 7,000 queries and 3,000 queries respectively. Table 8 shows the process of case C during the ten-day detection. As Table 8 shows, on the first day, SVM HYBRID starts with SS: the SVM margin covers 450 queries in the unknown subset of SS, of which 351 queries fall into the confusing region. Then, K -medoids clustering is performed on the 351 queries and obtains 70 suspicions. Afterwards, SVM HYBRID switches to ES on the unknown subset for ES and obtains 80 exemplars. Obtained suspicions and exemplars are manually labeled, and added to the training pool to update the detection model. This process repeats for the remaining days.

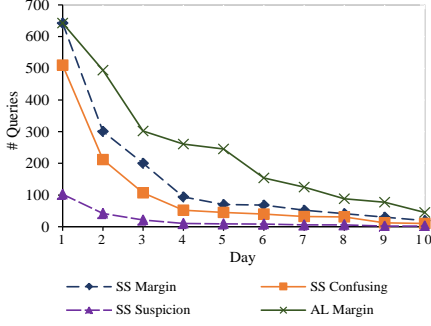


Figure 10: SS vs. SVM AL

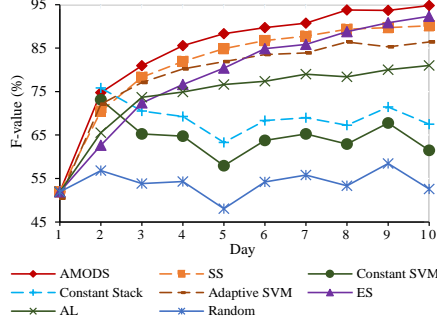


Figure 11: Comparisons of F-values

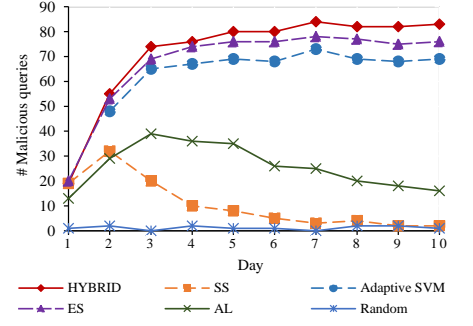


Figure 12: # Malicious queries obtained

Truly malicious queries obtained by our system can be used to update the WAF signature library. Hence, we expect that our system could maintain a desirable detection performance and simultaneously obtain as many malicious queries as possible. Figure 9 presents the trends of the number of malicious queries obtained by the five cases. It is clear that they all show an increasing trend. This is attributable to the fact that the number of exemplars increase while suspicions decline during the adaptive detection process, as concluded from Table 6. Suspicions are informative queries, yet exemplars are representative malicious queries. Hence, since exemplars dominate the total malicious queries obtained, the increase in the number of exemplars results in the increase in the number of total malicious queries. Last but not least, case C obtains the largest number of malicious queries among the five cases, not only on the last day but for the entire ten days. This confirms that suspicions help improve both the detection ability and the number of correctly classified exemplars, especially when θ is close to 7:3.

In summary, all the cases of AMODS yield desirable performance. The experiment results demonstrate the contribution of SVM HYBRID to AMODS: SS mainly helps improve the detection performance, while ES mainly helps obtain malicious queries. Since the ultimate goal of our system is to achieve a good detection performance and simultaneously obtain as many unknown malicious queries as possible, case C is chosen as the optimal case and 7:3 as the optimal value for θ , which slightly outperforms other cases in the two respects.

5.3. Experiment 3: comparisons of constant models and adaptive models

Experiment 3 aims to compare AMODS (case C) with constant models and adaptive models. Constant models are trained only once on the superset of the initial query set and the first-day unknown query set, including the constant SVM model and the constant stacking model. Five adaptive models are also tested: an adaptive model combining both stacking and SVM HYBRID in our proposed AMODS, an adaptive SVM model, SS, ES, random selection, and SVM AL, which is an existing effective adaptive

learning method (see Section 3.2.1). All the models in Experiment 3 are implemented in AMODS to carry out the ten-day detection with same predefined parameters as in Experiment 2, except that SS and SVM AL are required to obtain at most 150 important queries on each day, namely, $M \leq 150$.

Figure 10 depicts the process of SS in regard to the number of queries within the SVM margin (SS Margin), within the confusing region (SS Confusing), and obtained as suspicions (SS Suspicion) separately. SVM AL is also shown about the number of queries within the margin (AL Margin). They all present a decreasing trend. In comparison with SVM AL, important queries obtained by SS (SS Suspicion) are constantly fewer than those obtained by SVM AL (the smaller of AL Margin and 150). However, queries within the margin in SS are always fewer than those in SVM AL, which means the uncertainty of the detection model in SS is smaller than that in SVM AL. This phenomenon implies that SS is superior to SVM AL in reducing classifier uncertainty.

F-values of the eight detection approaches are exhibited in Figure 11. Among the five methods, random selection shows the worst performance and constantly fluctuates. The failure of random selection is attributed to the trivial probability of a randomly selected query being informative or representative. The constant SVM model and the constant stacking model perform better only than random selection; similar trends of the three methods also indicate that randomly selected queries make little contribution to detection performance improvement. Yet, for the five adaptive methods, steep uptrend is shown for the first three days and then steady uptrend for the remaining days. Detection performance improvement of AMODS over constant methods suggests that AMODS is able to effectively leverage existing knowledge, including knowledge gained during the adaptive detection process, toward enhanced subsequent detection of queries. Moreover, AMODS along with SS outperforms other adaptive methods. The improvement of AMODS over the adaptive SVM model implies the stronger classification ability of stacking over single classifiers. The superiority of AMODS mainly lies in that it employs SVM HYBRID, which combines the contributions of SS and ES while also mitigating their limita-

Table 9: Comparisons of web attack detection methods

Method	F-value(%)	TP rate (%)	FP rate (%)
Linear combination [10]	71.73	92.39	0.61
Wavelet transform [14]	61.04	82.61	0.81
Dimensionality reduction [15]	70.48	86.96	0.56
Adaptive learning (AMODS)	94.79	98.91	0.09

tions. Both SS and ES boost the performance of AMODS. Furthermore, SS outperforms ES, which demonstrates that suspicions make a valuable contribution to an accurate detection model, as validated in Figure 7. SS also outperforms SVM AL, even though SS obtains fewer important queries, as shown in Figure 10. This phenomenon reveals the advantage of SS over SVM AL: SS requires less manual labeling work and achieves better detection performance. The main weakness of SVM AL is its ignorance of density distribution of unknown samples, which leads to sample bias, while SS preserves the density distribution by using K -medoids clustering to obtain suspicions.

Figure 12 compares the number of truly malicious queries obtained by the six adaptive detection methods. Random selection still performs the worst. AMODS, together with ES and the adaptive SVM model, outperforms other methods, with a total number of 715, 656 and 615 malicious queries obtained respectively. The adaptive SVM model lags behind among the top three methods, which demonstrates the strong detection ability of the stacking-based detection model. Moreover, the reason why ES falls short of AMODS lies in that SS benefits the detection performance of AMODS, which leads AMODS to obtain more correctly-classified exemplars than ES does. Generally, AMODS, ES and the adaptive SVM model show an uptrend, while SS and SVM AL show a downtrend. This distinction is due to the difference in their preference for queries. AMODS, ES and the adaptive SVM model prefer queries that are highly likely to be malicious, while SS and SVM AL prefer queries that the classifier is uncertain about, and thus might not be malicious. The reduction in uncertain queries leads to a decline in malicious queries obtained by SS and SVM AL. During the ten-day detection, the total number of malicious queries obtained by AMODS (715) is 2.78 times and 59.58 times that by SVM AL (257) and random selection (12) respectively. This demonstrates the overwhelming ability of AMODS in harvesting malicious queries.

In conclusion, Experiment 3 demonstrates that adaptive learning can further enhance malicious query detection performance over constant models. Experiment results reveal the advantages of AMODS, SS and ES over SVM AL, constant models and random selection in two respects: improving detection performance and obtaining more malicious queries. AMODS wins in both respects among all the methods. SS does well in the former respect, while ES excels in the latter. Both SS and ES defeat other methods except for AMODS in their respective strengths.

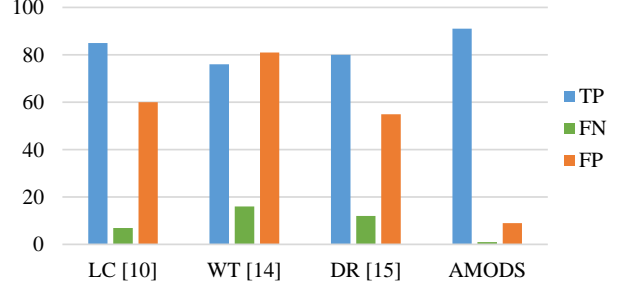


Figure 13: Comparisons of web attack detection methods

5.4. Experiment 4: comparisons with related work

In the context of web attack detection, existing methods are all constant and thus cannot adapt to the change of malicious behaviors. Our proposed adaptive detection method is capable of detecting the latest attacks by incrementally updating the detection model.

Experiment 4 aims at comparing AMODS with existing detection methods. However, our proposed method cannot be directly compared with them, since there is no public data available for web attack detection and existing works all used their own datasets for evaluation. The only labeled, publicly available trace in intrusion detection is DARPA 1999 [53], whose HTTP traffic is not appropriate for web attack detection, because it does not contain many web attacks [54]. For this reason, we use our dataset to test existing web attack detection methods, including linear combination [10], wavelet transform [14] and dimensionality reduction [15].

We set the anomaly threshold of [10] to the default 10% larger than the maximum anomaly score seen during training. [10] is conducted on our ten unknown sets within 10-fold cross validation. Since [10] uses normal traffic to build the detection model, malicious queries in the training set of each fold are removed. [14] and [15] are unsupervised, so do not require a training phase. We set their anomaly threshold to 0.92%, the percentage of malicious queries in our dataset. TW2D-BA is used to perform [14], while PCA along with RM is used to implement [15]. [14] and [15] are implemented on each unknown dataset, and their performance is averaged separately.

Table 9 and Figure 13 compare the detection performance of AMODS with the three web attack detection methods. The comparison results illustrate that our method is the overall winner and significantly beats the other methods, achieving the highest F-value (94.79%) and

TP rate (98.91%), as well as the lowest FP rate (0.09%) among them. Among the three methods, [10] obtains the highest F-value (71.73%) and the highest TP rate (92.39%). However, they are still lower than the lowest F-value and TP rate obtained by AMODS on the 10th day, namely 91.67% and 95.65%. Moreover, their lowest FP rate (0.56%) is higher than the highest of SVM HYBRID on the 10th day (0.17%).

We attribute the success of AMODS to the principle of SVM HYBRID, namely, choosing both the most important prone-to-be-misclassified queries (suspicions) and the most representative malicious queries (exemplars) from unknown queries. By incrementally incorporating correctly-labeled suspicions and exemplars into the training pool to update the detection model, its detection ability is greatly enhanced. The detection model is up-to-date and capable of detecting the most recent unknown attacks.

6. Discussion

In this paper, we propose an adaptive learning strategy called SVM HYBRID for the efficient selection of important queries, and develop a malicious query detection system that can adaptively update the detection model with a small manual labeling workload. In this section, we clarify some relevant issues of our approach and discuss the limitations of our approach.

Labeling unknown queries. Manually labeling the 112,397 queries in web requests is time-consuming and error-prone. Therefore, we initially used the rules of ModSecurity to detect attacks automatically. Then we manually confirmed the detected attacks, and concluded that the TP rate and FP rate of the detection results was 33.37% and 8.64%. To decipher the reason for the unsatisfactory detection results, we manually checked the ModSecurity rule set and found that some rules matched queries with simple malicious code snippets, which might appear in normal queries, resulting in false positives. Malicious queries that contain rare or complex malicious code snippets do not match any rules, and might evade the detection, leading to false negatives. This demonstrates the necessity of updating the WAF signature library, which can be achieved by acquiring the latest malicious queries via our system.

Code obfuscation. As mentioned above, we labeled unknown queries using static analysis, which might be bypassed by attacks using code obfuscation. Code obfuscation could deteriorate the detection performance of static techniques, so we needed to eliminate the impact on AMODS. To this end, we assumed that queries with obfuscated code were malicious and made no further analysis on them. This operation is rational, because the reason for obfuscation in web requests is to evade detection, and it is futile to obfuscate benign web requests. If necessary, dynamic analysis could be applied to defend against code obfuscation. Dynamic techniques can monitor the code

that is actually executed by the application, and thus are immune to many code-obfuscating transformations.

Web application concept drift. In machine learning, changes in the modeled behaviors of an observed object are known as concept drift [55]. Maggi et al. [56] defined the behaviors of a web application as functionalities that the application offers and, more specifically, the content of web requests and responses. They also defined the significant changes that frequently happen to behaviors of a web application as web application concept drift. Web application concept drift is common nowadays because web applications become increasingly multi-featured to satisfy users' needs. If web application concept drift is unknown to the web attack detection model, the model might misclassify the latest normal behaviors of the application after an update, leading to false positives. Therefore, by keeping track of the FP rate of the daily-updated detection model on the original ten-day query set, our system could also be used to sense concept drift. The FP rate increased dramatically on queries on the 6th day; therefore it is likely that the web application concept drift occurred on this day. We confirmed our speculation by analyzing logs and contacting the site manager. The website was upgraded on the 6th day: a new category for news was established, which caused the query structure in requests to the links in the new category to change, with new parameter names.

Future work. For future research, we intend to apply deep learning to discover more previously unknown attack features to replace N -gram features, since N -grams struggle to interpret attack patterns. We also plan to enable AMODS to perform in a fully automatic manner without human supervision while maintaining its high detection rate, though at present it only requires a trivial amount of manual labeling work: 0.15% (150 out of 100,000) of the entire unknown queries. An intuitive solution is unsupervised learning. Beyond that, AMODS is currently limited to detecting malicious queries in web attacks. Nevertheless, it has excellent potential for use in network intrusion detection. Comprehensive evaluation will be conducted to capture the performance of AMODS in this area.

7. Conclusion

Queries in web requests can be exploited by attackers to conduct attacks against web servers and web applications. This paper introduces a novel adaptive system for detecting malicious queries, called AMODS. It is, to the best of our knowledge, the first research into adaptive detection in the field of web attack detection. AMODS takes daily web traffic as input, and models queries in web requests. The core of AMODS, named SVM HYBRID, an adaptive learning approach, serves to reduce manual work by choosing important queries for labeling. SVM HYBRID is a hybrid of Suspicion Selection (SS) and Exemplar Selection (ES). SS obtains the most important informative queries and features in improving detection performance, while ES specializes in harvesting truly malicious queries.

Queries obtained by SVM HYBRID are incrementally incorporated into the training pool to update the detection model. Malicious queries obtained could also be used to update the WAF signature library. AMODS was tested on the real-world ten-day query set of an academic institute. Experiments show that AMODS outperforms SVM AL in terms of both the detection performance and the number of malicious queries obtained. AMODS also leads existing web attack detection methods on our dataset with the highest F-value of 94.79% and the lowest FP-rate of 0.09%.

Acknowledgement

This work is supported in part by the National Natural Science Foundation of China (61272481, 61303239, 61572460), the National key research and development project (2016YFB0800703), the National Information Security Special Projects of National Development and Reform Commission of China [(2012)1424], open Project Program of the State Key Laboratory of Information Security(2016-MS-02). I would also like to express my very great appreciation to the assistant professor Xinyu Xing in Pennsylvania State University for his valuable and constructive suggestions during the development of this research work.

References

- [1] Symantec, Internet security threat report 2016, <https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf>.
- [2] J. Fonseca, M. Vieira, H. Madeira, Evaluation of web security mechanisms using vulnerability & attack injection, Dependable & Secure Computing IEEE Transactions on 11 (5) (2014) 440–453.
- [3] Imperva, Web application attack report 2014, https://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed6.pdf.
- [4] Imperva, Web application attack report 2015, https://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed6.pdf.
- [5] Imperva, Web application attack report 2016, https://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed6.pdf.
- [6] M. Lawal, A. B. M. Sultan, A. O. Shakiru, Systematic literature review on sql injection attack, International Journal of Soft Computing 11 (1) (2016) 26–35.
- [7] Symantec, Team ghostshell hacking group back with a bang, <https://www.symantec.com/connect/blogs/team-ghostshell-hacking-group-back-bang>.
- [8] I. Hydera, A. B. M. Sultan, H. Zulzalil, N. Admodisastro, Current state of research on cross-site scripting (xss)—a systematic literature review, Information and Software Technology 58 (2015) 170–186.
- [9] V. Prokhorenko, K. K. R. Choo, H. Ashman, Web application protection techniques: A taxonomy, Journal of Network & Computer Applications 60 (2016) 95–112.
- [10] C. Kruegel, G. Vigna, Anomaly detection of web-based attacks, in: ACM Conference on Computer and Communications Security, 2003, pp. 251 – 261.
- [11] C. Kruegel, G. Vigna, W. Robertson, A multi-model approach to the detection of web-based attacks, Computer Networks the International Journal of Computer & Telecommunications Networking 48 (5) (2005) 717–738.
- [12] W. K. Robertson, G. Vigna, C. Kruegel, R. A. Kemmerer, Using generalization and characterization techniques in the anomaly-based detection of web attacks., in: Network and Distributed System Security Symposium, NDSS 2006, San Diego, California, Usa, 2006.
- [13] Y. Song, A. D. Keromytis, S. J. Stolfo, Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic., in: Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, Usa, February - February, 2009, pp. 121–135.
- [14] A. Kozakevicius, C. Cappel, B. A. Mozzaquatro, R. C. Nunes, C. E. Schaerer, Url query string anomaly sensor designed with the bidimensional haar wavelet transform, International Journal of Information Security 14 (6) (2015) 561–581.
- [15] A. Juvonen, T. Sipola, T. Inen, Online anomaly detection using dimensionality reduction techniques for http log analysis, Computer Networks the International Journal of Computer & Telecommunications Networking 91 (C) (2015) 46–56.
- [16] Y. Meng, L. F. Kwok, Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection, Journal of Network & Computer Applications 39 (1) (2014) 83–92.
- [17] W. Wang, T. Guyet, R. Quiniou, M. O. Cordier, F. Massegia, X. Zhang, Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks, Knowledge-Based Systems 70 (2014) 103–117.
- [18] J. Zhang, H. Li, Q. Gao, H. Wang, Y. Luo, Detecting anomalies from big network traffic data using an adaptive detection approach, Information Sciences An International Journal 318 (C) (2015) 91–110.
- [19] V. Vapnik, Estimation of Dependences Based on Empirical Data, Springer New York, 2006.
- [20] P. Brazdil, C. G. Carrier, C. Soares, R. Vilalta, Metalearning: Applications to data mining, Springer Science & Business Media, 2008.
- [21] Y. Zheng, X. Zhang, Path sensitive static analysis of web applications for remote code execution vulnerability detection, in: International Conference on Software Engineering, 2013, pp. 652–661.
- [22] P. Garcia-Teodoro, J. E. Diaz-Verdejo, J. E. Tapiador, R. Salazar-Hernandez, Automatic generation of http intrusion signatures by selective identification of anomalies, Computers & Security 55 (C) (2015) 159–174.
- [23] Y. Zhong, H. Asakura, H. Takakura, Y. Oshima, Detecting malicious inputs of web application parameters using character class sequences, in: Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual, Vol. 2, IEEE, 2015, pp. 525–532.
- [24] D. Ariu, R. Tronci, G. Giacinto, Hmmpayl : An intrusion detection system based on hidden markov models, Computers & Security 30 (4) (2011) 221–241.
- [25] K. Wang, S. J. Stolfo, Anomalous payload-based network intrusion detection, Lecture Notes in Computer Science 3224 (2004) 203–222.
- [26] K. Wang, J. J. Parekh, S. J. Stolfo, Anagram: a content anomaly detector resistant to mimicry attack, Springer Berlin Heidelberg, 2006.
- [27] A. Jamdagni, Z. Tan, X. He, P. Nanda, R. P. Liu, Repids: A multi tier real-time payload-based intrusion detection system, Computer Networks 57 (3) (2013) 811–824.
- [28] A. Oza, K. Ross, R. M. Low, M. Stamp, Http attack detection using n -gram analysis, Computers & Security 45 (3) (2014) 242–254.
- [29] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, W. Lee, Mcpad: A multiple classifier system for accurate payload-based anomaly detection, Computer Networks the International Journal of Computer & Telecommunications Networking 53 (6) (2009) 864–881.
- [30] M. Swarnkar, N. Hubballi, Ocpad: One class naive bayes classi-

- fier for payload based anomaly detection, *Expert Systems with Applications* 64 (2016) 330–339.
- [31] J. Dahse, T. Holz, Static detection of second-order vulnerabilities in web applications.
- [32] B. Livshits, S. Chong, Towards fully automatic placement of security sanitizers and declassifiers, *Acm Sigplan Notices* 48 (48) (2010) 385–398.
- [33] A. Naderi-Afooshteh, A. Nguyen-Tuong, M. Bagheri-Marzijarani, J. D. Hiser, J. W. Davidson, Joza: Hybrid taint inference for defeating web application sql injection attacks, in: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE, 2015, pp. 172–183.
- [34] G. M. Howard, C. N. Gutierrez, F. A. Arshad, S. Bagchi, Y. Qi, psigene: Webcrawling to generalize sql injection signatures, in: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE, 2014, pp. 45–56.
- [35] G. Schohn, D. Cohn, Less is more: Active learning with support vector machines, in: *Int Conf*, 2000, pp. 839–846.
- [36] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, *Journal of Machine Learning Research* 2 (1) (2001) 45–66.
- [37] B. R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis*, ch, Wiley, 1973.
- [38] Y. Baram, E. Y. Ran, K. Luz, Online choice of active learning algorithms., *Journal of Machine Learning Research* 5 (1) (2012) 255–291.
- [39] F. Sebastiani, Machine learning in automated text categorization, *Acm Computing Surveys* 34 (1) (2002) 1–47.
- [40] L. Rendell, R. Sheshu, D. Tcheng, Layered concept-learning and dynamically variable bias management, in: *International Joint Conference on Artificial Intelligence*, 1987, pp. 308–314.
- [41] L. Rokach, Ensemble-based classifiers, *Artificial Intelligence Review* 33 (1-2) (2010) 1–39.
- [42] D. H. Wolpert, Stacked generalization *, *Neural Networks* 5 (2) (1992) 241–259.
- [43] H. Networks, Hillstone e-series next-generation firewalls, <http://www.hillstonenet.com/>.
- [44] Trustwave, Modsecurity, <https://www.modsecurity.org/>.
- [45] Trustwave, Modsecurity core rule set, https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project.
- [46] S. Prandl, M. Lazarescu, D. S. Pham, *A Study of Web Application Firewall Solutions*, Springer International Publishing, 2015.
- [47] R. Fielding, G. S. J., J. Mogul, H. Nielsen, L. Masinter, P. Leach, T. Berners-Lee, Rfc 2616: Hypertext transfer protocol - http/1.1, *Computer Science & Communications Dictionary* 7 (9) (1999) 3969 – 3973.
- [48] W. B. Cavnar, J. M. Trenkle, et al., N-gram-based text categorization, *Ann Arbor MI* 48113 (2) (1994) 161–175.
- [49] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, S. J. Cunningham, *Weka: Practical machine learning tools and techniques with java implementations*.
- [50] C. C. Chang, C. J. Lin, Libsvm: A library for support vector machines, *Acm Transactions on Intelligent Systems & Technology* 2 (3, article 27) (2007) 389–396.
- [51] J. Han, J. Pei, M. Kamber, *Data mining: concepts and techniques*, Elsevier, 2011.
- [52] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al., *A practical guide to support vector classification*.
- [53] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, K. Das, The 1999 darpa off-line intrusion detection evaluation, *Computer networks* 34 (4) (2000) 579–595.
- [54] J. Mchugh, Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, *Acm Transactions on Information & System Security* 3 (4) (2000) 262–294.
- [55] J. C. Schlimmer, R. H. Granger, Beyond incremental processing: Tracking concept drift., in: *National Conference on Artificial Intelligence*. Philadelphia, Pa, August 11-15, 1986. Volume 1: Science, 1986, pp. 502–507.
- [56] F. Maggi, W. Robertson, C. Kruegel, G. Vigna, Protecting a moving target: Addressing web application concept drift, in: *International Symposium on Recent Advances in Intrusion Detection*, 2009, pp. 21–40.